



NQ Library Reference

YNQ
YNQ-1.6.0

Contents

1	Module Index	1
1.1	Modules	1
2	File Index	2
2.1	File List	2
3	Module Documentation	3
3.1	System Dependent	3
3.1.1	Detailed Description	3
3.2	Time	4
3.2.1	Detailed Description	4
3.2.2	Macro Definition Documentation	4
3.2.2.1	syGetTimeInSec	4
3.2.2.2	sySleep	4
3.2.2.3	syUSleep	4
3.2.3	Function Documentation	4
3.2.3.1	syGetTimeInMsec	4
3.2.3.2	syConvertTimeSpecToTimeInMsec	5
3.2.3.3	syConvertTimeInMsecToSec	6
3.2.3.4	syGetTimeZone	6
3.2.3.5	syDecomposeTime	6
3.2.3.6	syComposeTime	6
3.2.3.7	syGmtToString	6
3.3	Threads	8
3.3.1	Detailed Description	8
3.3.2	Macro Definition Documentation	8
3.3.2.1	SYThread	8
3.3.2.2	syIsValidThread	8
3.3.2.3	syThreadGetCurrent	8
3.3.2.4	syThreadDestroy	8
3.3.3	Function Documentation	8
3.3.3.1	syThreadStart	8
3.4	Mutexes	9
3.4.1	Detailed Description	9
3.4.2	Macro Definition Documentation	9
3.4.2.1	SYMutex	9
3.4.2.2	syMutexTake	9
3.4.3	Function Documentation	9
3.4.3.1	syMutexCreate	9
3.4.3.2	syMutexDelete	9
3.4.3.3	syMutexTakeDebug	9
3.4.3.4	syMutexGive	10
3.5	Semaphores	11
3.5.1	Detailed Description	11
3.5.2	Macro Definition Documentation	11
3.5.2.1	SY_SEMAPHORE_AVAILABLE	11
3.5.2.2	SYSemaphore	11
3.5.2.3	sySemaphoreDelete	11

3.5.2.4	sySemaphoreTake	11
3.5.2.5	sySemaphoreGive	11
3.5.3	Function Documentation	11
3.5.3.1	sySemaphoreCreate	11
3.5.3.2	sySemaphoreResetCounter	12
3.5.3.3	sySemaphoreTimedTake	12
3.6	Network	13
3.6.1	Detailed Description	13
3.6.2	Macro Definition Documentation	13
3.6.2.1	SY_LOCALHOSTIP4	13
3.6.2.2	SY_LOCALHOSTIP6	13
3.6.2.3	SY_LINKLOCALIP	13
3.6.2.4	SY_ANYIP4	13
3.6.2.5	SY_ANYIP6	13
3.6.2.6	SY_ZEROIP	13
3.6.2.7	SY_ZEROIP4	13
3.6.2.8	syGetHostName	13
3.6.3	Function Documentation	13
3.6.3.1	syGetIPv6Scopeld	13
3.6.3.2	syGetHostByName	14
3.6.3.3	syGetDnsParams	14
3.6.3.4	syGetMacAddress	14
3.6.3.5	syGetAdapter	14
3.7	Sockets	16
3.7.1	Detailed Description	16
3.7.2	Macro Definition Documentation	17
3.7.2.1	SY_INTERNALSOCKETPOOL	17
3.7.2.2	SYSocketHandle	17
3.7.2.3	SYSocketSet	17
3.7.2.4	syIsValidSocket	17
3.7.2.5	syInvalidSocket	17
3.7.2.6	syAddSocketToSet	17
3.7.2.7	syIsSocketSet	17
3.7.2.8	syClearSocketSet	17
3.7.2.9	syClearSocketFromSet	17
3.7.2.10	sySetDatagramSocketOptions	17
3.7.2.11	sySetStreamSocketOptions	17
3.7.3	Function Documentation	17
3.7.3.1	syIsSocketAlive	17
3.7.3.2	syShutdownSocket	17
3.7.3.3	syCloseSocket	18
3.7.3.4	syListenSocket	18
3.7.3.5	syCreateSocket	18
3.7.3.6	syBindSocket	18
3.7.3.7	syAllowBroadcastsSocket	19
3.7.3.8	sySetClientSocketOptions	19
3.7.3.9	syGetSocketPortAndIP	19
3.7.3.10	sySendToSocket	19
3.7.3.11	syConnectSocket	20
3.7.3.12	sySendSocket	20
3.7.3.13	sySendSocketAsync	20
3.7.3.14	sySelectSocket	21
3.7.3.15	syRecvFromSocket	22
3.7.3.16	syRecvSocket	22
3.7.3.17	syRecvSocketWithTimeout	22
3.7.3.18	syAcceptSocket	23
3.7.3.19	sySendMulticast	23
3.7.3.20	sySubscribeToMulticast	23

3.8	Tasks	24
3.8.1	Detailed Description	24
3.8.2	Macro Definition Documentation	24
3.8.2.1	syGetPid	24
3.9	Directories	25
3.9.1	Detailed Description	25
3.9.2	Macro Definition Documentation	25
3.9.2.1	SYDirectory	25
3.9.2.2	syInvalidateDirectory	25
3.9.2.3	syIsValidDirectory	25
3.9.3	Function Documentation	25
3.9.3.1	syCreateDirectory	25
3.9.3.2	syDeleteDirectory	25
3.9.3.3	syOpenDirectory	26
3.9.3.4	syFirstDirectoryFile	27
3.9.3.5	syNextDirectoryFile	27
3.9.3.6	syCloseDirectory	27
3.10	Files	28
3.10.1	Detailed Description	28
3.10.2	Macro Definition Documentation	28
3.10.2.1	SY_EXTENDFILENOTSUPPORTED	28
3.10.2.2	SY_PATHSEPARATOR	28
3.10.2.3	SYFile	29
3.10.2.4	syInvalidateFile	29
3.10.2.5	syIsValidFile	29
3.10.2.6	syInvalidFile	29
3.10.2.7	SY_CP_FIRSTILLEGALCHAR	29
3.10.2.8	SY_CP_ANYILLEGALCHAR	29
3.10.2.9	syFlushFile	29
3.10.2.10	syGetSecurityDescriptor	29
3.10.2.11	sySetSecurityDescriptor	29
3.10.3	Function Documentation	29
3.10.3.1	syCreateFile	29
3.10.3.2	syDeleteFile	30
3.10.3.3	syRenameFile	30
3.10.3.4	syOpenFileForRead	30
3.10.3.5	syOpenFileForWrite	30
3.10.3.6	syOpenFileForReadWrite	31
3.10.3.7	syTruncateFile	31
3.10.3.8	syReadFile	31
3.10.3.9	syWriteFile	31
3.10.3.10	syCloseFile	32
3.10.3.11	sySeekFileCurrent	32
3.10.3.12	sySeekFileStart	32
3.10.3.13	sySeekFileEnd	32
3.10.3.14	syGetFileInformation	33
3.10.3.15	syGetFileInformationByName	33
3.10.3.16	syGetFileSize	33
3.10.3.17	sySetFileInformation	33
3.10.3.18	syGetVolumeInformation	34
3.11	Input Output	35
3.11.1	Detailed Description	35
3.11.2	Macro Definition Documentation	35
3.11.2.1	syPrintf	35
3.11.2.2	syFprintf	35
3.11.2.3	sySprintf	35
3.11.2.4	sySnprintf	35
3.11.2.5	syVsnprintf	35

3.11.2.6	sySscanf	35
3.11.2.7	sySscanf	35
3.11.2.8	syGetchar	35
3.11.2.9	syFflush	35
3.12	File Locking	36
3.12.1	Detailed Description	36
3.12.2	Macro Definition Documentation	36
3.12.2.1	syUnlockFile	36
3.12.2.2	syLockFile	36
3.13	Direct Transfer	37
3.13.1	Detailed Description	37
3.13.2	Function Documentation	37
3.13.2.1	syDtStartPacket	37
3.13.2.2	syDtEndPacket	37
3.13.2.3	syDtFromSocket	37
3.13.2.4	syDtToSocket	37
3.14	Encoding	39
3.14.1	Detailed Description	39
3.14.2	Macro Definition Documentation	39
3.14.2.1	SY_UNICODEFILESYSTEM	39
3.14.3	Function Documentation	39
3.14.3.1	syUnicodeToUTF8N	39
3.14.3.2	syUTF8ToUnicodeN	39
3.14.3.3	initCodePageUTF8	39
3.15	Utils	41
3.15.1	Detailed Description	41
3.15.2	Macro Definition Documentation	41
3.15.2.1	sySetRand	41
3.15.2.2	syRand	41
3.15.3	Function Documentation	41
3.15.3.1	syUnixMode2DosAttr	41
3.15.3.2	syGetLastSmbError	42
3.15.3.3	sySetLastNqError	42
4	File Documentation	43
4.1	syopsyst.h File Reference	43
4.1.1	Macro Definition Documentation	46
4.1.1.1	MUTEX_DEBUG	46
4.1.1.2	SY_OSNAME	46
4.1.1.3	syAssert	46

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

System Dependent	3
Time	4
Threads	8
Mutexes	9
Semaphores	11
Network	13
Direct Transfer	37
Sockets	16
Tasks	24
Directories	25
Files	28
Input Output	35
File Locking	36
Encoding	39
Utils	41

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

syopsyst.h	43
--------------------------------------	----

Chapter 3

Module Documentation

3.1 System Dependent

Modules

- [Time](#)
- [Threads](#)
- [Mutexes](#)
- [Semaphores](#)
- [Network](#)
- [Sockets](#)
- [Tasks](#)
- [Directories](#)
- [Files](#)
- [Input Output](#)
- [File Locking](#)
- [Encoding](#)
- [Utils](#)

3.1.1 Detailed Description

3.2 Time

Macros

- #define [syGetTimeInSec\(\)](#) time(0)
- #define [sySleep\(_secs_\)](#) sleep(_secs_)
- #define [syUSleep\(_msecs_\)](#) usleep(_msecs_)

Functions

- NQ_TIME [syGetTimeInMsec](#) (void)
- NQ_TIME [syConvertTimeSpecToTimeInMsec](#) (void *val)
- NQ_UINT32 [syConvertTimeInMsecToSec](#) (NQ_TIME *timeMsec)
- NQ_INT [syGetTimeZone](#) (void)
- void [syDecomposeTime](#) (NQ_UINT32 time, SYTimeFragments *decomposed)
- NQ_UINT32 [syComposeTime](#) (const SYTimeFragments *decomposed)
- NQ_BOOL [syGmtToString](#) (NQ_BYTE *strTime, NQ_COUNT size, NQ_UINT32 t, const NQ_CHAR *fmt)

3.2.1 Detailed Description

Time functions

See Also

[syGetTimeAccuracy\(\)](#) in the platform-dependent sypltfm.h file

3.2.2 Macro Definition Documentation

3.2.2.1 #define syGetTimeInSec() time(0)

System (Posix) time in seconds from 1-Jan-1970

3.2.2.2 #define sySleep(_secs_) sleep(_secs_)

Wait a number of seconds.

3.2.2.3 #define syUSleep(_msecs_) usleep(_msecs_)

Wait a number of milliseconds.

3.2.3 Function Documentation

3.2.3.1 NQ_TIME syGetTimeInMsec (void)

Get system time in POSIX format in milliseconds

Returns

The number of milliseconds elapsed since Jan 1, 1970 in milliseconds

3.2.3.2 NQ_TIME syConvertTimeSpecToTimeInMsec (void * *val*)

Convert the **Timespec** to time in milliseconds

Parameters

<i>val</i>	Timespec value format
------------	------------------------------

Returns

The converted time from **Timespec** to a time in milliseconds

3.2.3.3 NQ_UINT32 syConvertTimeInMsecToSec (NQ_TIME * *timeMsec*)

Convert time in milliseconds to seconds

Parameters

<i>timeMsec</i>	Time in milliseconds.
-----------------	-----------------------

Returns

Time in seconds

3.2.3.4 NQ_INT syGetTimeZone (void)

Get time zone difference in minutes

Returns

The number of minutes to be added to the local time to get GMT. This number is negative for GMT+ and positive for GMT-

3.2.3.5 void syDecomposeTime (NQ_UINT32 *time*, SYTimeFragments * *decomposed*)

Decompose system time into fragments

Parameters

<i>time</i>	System time
<i>decomposed</i>	Structure of file fragments

3.2.3.6 NQ_UINT32 syComposeTime (const SYTimeFragments * *decomposed*)

Compose system time from fragments

Parameters

<i>decomposed</i>	Structure of file fragments
-------------------	-----------------------------

Returns

Composed system time

3.2.3.7 NQ_BOOL syGmtToString (NQ_BYTE * *strTime*, NQ_COUNT *size*, NQ_UINT32 *t*, const NQ_CHAR * *fmt*)

This function converts system time into GMT time and prints it according to the format (*fmt*).

GMT-Y.m.d-H.M.S

Parameters

<i>strTime</i>	The buffer to return the string.
<i>size</i>	The buffer size.
<i>t</i>	Time in seconds from Jan 1, 1970 (Unix time).
<i>fmt</i>	The string format.

Returns

TRUE on success, *FALSE* on error.

Note

The format string in the example is currently the only one that need be supported.

3.3 Threads

Macros

- #define SYThread pthread_t
- #define sylvValidThread(_taskId_) TRUE
- #define syThreadGetCurrent pthread_self
- #define syThreadDestroy(_taskId_) pthread_cancel(_taskId_);

Functions

- void syThreadStart (NQ_BOOL isRT, NQ_INT priorityLevel, SYThread *taskIdPtr, void(*startpoint)(void), NQ_BOOL background)

3.3.1 Detailed Description

Threads

Thread management calls.

3.3.2 Macro Definition Documentation

3.3.2.1 #define SYThread pthread_t

TID - thread handle

3.3.2.2 #define sylvValidThread(_taskId_) TRUE

3.3.2.3 #define syThreadGetCurrent pthread_self

3.3.2.4 #define syThreadDestroy(_taskId_) pthread_cancel(_taskId_);

3.3.3 Function Documentation

3.3.3.1 void syThreadStart (NQ_BOOL isRT, NQ_INT priorityLevel, SYThread * taskIdPtr, void(*) (void) startpoint, NQ_BOOL background)

Create an internal thread

Parameters

<i>isRT</i>	Whether to set priority level according to priorityLevel
<i>priorityLevel</i>	The priority level of the thread
<i>taskIdPtr</i>	Pointer to a thread handle to create
<i>startpoint</i>	Pointer to the function to be used as the thread startpoint
<i>background</i>	TRUE if the thread should have low priority. FALSE if the thread should have normal priority.

3.4 Mutexes

Macros

- #define [SYMutex](#) pthread_mutex_t
- #define [syMutexTake](#)(_m) [syMutexTakeDebug](#)(_m, SY_LOG_FILE, SY_LOG_LINE)

Functions

- void [syMutexCreate](#) (SYMutex *_m)
- void [syMutexDelete](#) (SYMutex *_m)
- void [syMutexTakeDebug](#) (SYMutex *_m, const NQ_CHAR *text, const NQ_UINT line)
- void [syMutexGive](#) (SYMutex *_m)

3.4.1 Detailed Description

3.4.2 Macro Definition Documentation

3.4.2.1 #define SYMutex pthread_mutex_t

3.4.2.2 #define syMutexTake(_m) syMutexTakeDebug(_m, SY_LOG_FILE, SY_LOG_LINE)

3.4.3 Function Documentation

3.4.3.1 void syMutexCreate (SYMutex *_m)

Create a mutual exception semaphore

Parameters

<code>_m</code>	Pointer to a semaphore variable to create
-----------------	---

3.4.3.2 void syMutexDelete (SYMutex *_m)

Dispose a mutual exception semaphore

Parameters

<code>_m</code>	Pointer to a semaphore variable to dispose
-----------------	--

3.4.3.3 void syMutexTakeDebug (SYMutex *_m, const NQ_CHAR * *text*, const NQ_UINT *line*)

Lock a resource protected by a mutex

Parameters

<code>_m</code>	Pointer to a mutex semaphore
-----------------	------------------------------

<i>text</i>	File name for debug
<i>line</i>	Line number for debug

3.4.3.4 void syMutexGive (SYMutex * *_m*)

Unlock a resource protected by a mutex

Parameters

<i>_m</i>	Pointer to a mutex semaphore
-----------	------------------------------

3.5 Semaphores

Macros

- #define SY_SEMAPHORE_AVAILABLE
- #define SYSemaphore sem_t
- #define sySemaphoreDelete(_s) sem_destroy(&_s)
- #define sySemaphoreTake(_s) sem_wait(&_s)
- #define sySemaphoreGive(_s) sem_post(&_s)

Functions

- NQ_STATUS sySemaphoreCreate (SYSemaphore *semId, NQ_UINT count)
- void sySemaphoreResetCounter (SYSemaphore *pSemId)
- NQ_INT sySemaphoreTimedTake (SYSemaphore *sem, NQ_INT timeout)

3.5.1 Detailed Description

Semaphores

1) We use mutex semaphores or simulate their behavior if the OS does not support pure mutex semaphores. 2) We use binary semaphores

3.5.2 Macro Definition Documentation

3.5.2.1 #define SY_SEMAPHORE_AVAILABLE

Enable this define when semaphores are supported on system

3.5.2.2 #define SYSemaphore sem_t

Counting semaphore

3.5.2.3 #define sySemaphoreDelete(_s) sem_destroy(&_s)

3.5.2.4 #define sySemaphoreTake(_s) sem_wait(&_s)

3.5.2.5 #define sySemaphoreGive(_s) sem_post(&_s)

3.5.3 Function Documentation

3.5.3.1 NQ_STATUS sySemaphoreCreate (SYSemaphore * *semId*, NQ_UINT *count*)

Create semaphore

Parameters

<i>semId</i>	Pointer to the semaphore id
--------------	-----------------------------

<i>count</i>	Number of resources
--------------	---------------------

Returns

NQ_SUCCESS when semaphore was created or NQ_FAIL on error

3.5.3.2 void sySemaphoreResetCounter (SYSemaphore * *pSemID*)

Reset semaphore counter

Parameters

<i>pSemID</i>	Pointer to the semaphore id
---------------	-----------------------------

3.5.3.3 NQ_INT sySemaphoreTimedTake (SYSemaphore * *sem*, NQ_INT *timeout*)

Lock a resource protected by a binary semaphore

Parameters

<i>sem</i>	Binary semaphore
<i>timeout</i>	Timeout in seconds to wait for lock

Returns

NQ_SUCCESS when the resource was locked or NQ_FAIL on error

3.6 Network

Modules

- [Direct Transfer](#)

Macros

- #define [SY_LOCALHOSTIP4](#) /*{0, 0} */ {0x7f00 , 0x0001}
- #define [SY_LOCALHOSTIP6](#) {0, 0, 0, 0, 0, 0, 0, 1} /* {0, 0, 0, 0, 0, 0, 0, 0} */
- #define [SY_LINKLOCALIP](#) 0xfe80
- #define [SY_ANYIP4](#) {0, 0}
- #define [SY_ANYIP6](#) {0, 0, 0, 0, 0, 0, 0, 0}
- #define [SY_ZEROIP](#) {0, 0, 0, 0, 0, 0, 0, 0}
- #define [SY_ZEROIP4](#) 0L
- #define [syGetHostName](#)(*_name*, *_nameLen*) gethostname(*(_name)*, *(_nameLen)*)

Functions

- NQ_UINT32 [syGetIPv6ScopeId](#) (const NQ_IPADDRESS6 ip)
- NQ_IPADDRESS4 [syGetHostByName](#) (const char *name)
- void [syGetDnsParams](#) (NQ_CHAR *domain, NQ_IPADDRESS *server)
- void [syGetMacAddress](#) (NQ_IPADDRESS4 ip, NQ_BYTE *macBuffer)
- NQ_STATUS [syGetAdapter](#) (NQ_INDEX adapterIdx, NQ_INDEX *osIndex, NQ_IPADDRESS4 *pIp, NQ_IPADDRESS6 *pIp6, NQ_IPADDRESS4 *pSubnet, NQ_IPADDRESS4 *pBcast, NQ_WCHAR *pWins, NQ_WCHAR *pDns)

3.6.1 Detailed Description

3.6.2 Macro Definition Documentation

3.6.2.1 #define SY_LOCALHOSTIP4 /*{0, 0} */ {0x7f00 , 0x0001}

3.6.2.2 #define SY_LOCALHOSTIP6 {0, 0, 0, 0, 0, 0, 0, 1} /* {0, 0, 0, 0, 0, 0, 0, 0} */

3.6.2.3 #define SY_LINKLOCALIP 0xfe80

3.6.2.4 #define SY_ANYIP4 {0, 0}

3.6.2.5 #define SY_ANYIP6 {0, 0, 0, 0, 0, 0, 0, 0}

3.6.2.6 #define SY_ZEROIP {0, 0, 0, 0, 0, 0, 0, 0}

3.6.2.7 #define SY_ZEROIP4 0L

3.6.2.8 #define syGetHostName(*_name*, *_nameLen*) gethostname(*(_name)*, *(_nameLen)*)

3.6.3 Function Documentation

3.6.3.1 NQ_UINT32 syGetIPv6ScopeId (const NQ_IPADDRESS6 ip)

Get IPv6 scope ID

Parameters

<i>ip</i>	The IPv6 address
-----------	------------------

Returns

0..n - The scope id to be used

Note

0 - Unknown network interface

3.6.3.2 NQ_IPADDRESS4 syGetHostByName (const char * *name*)

Find host IP by its name

Parameters

<i>name</i>	Host name
-------------	-----------

Returns

Host IP

3.6.3.3 void syGetDnsParams (NQ_CHAR * *domain*, NQ_IPADDRESS * *server*)

Returns the DNS initializations parameters

Parameters

<i>domain</i>	The default domain target belongs to
<i>server</i>	The DNS server IP address

3.6.3.4 void syGetMacAddress (NQ_IPADDRESS4 *ip*, NQ_BYTE * *macBuffer*)

Get MAC address by IP4

Parameters

<i>ip</i>	Next IP address
<i>macBuffer</i>	Buffer for mac address

3.6.3.5 NQ_STATUS syGetAdapter (NQ_INDEX *adapterIdx*, NQ_INDEX * *osIndex*, NQ_IPADDRESS4 * *pIp*, NQ_IPADDRESS6 * *ip6*, NQ_IPADDRESS4 * *pSubnet*, NQ_IPADDRESS4 * *pBcast*, NQ_WCHAR * *pWins*, NQ_WCHAR * *pDns*)

Get adapter information

Parameters

<i>adapterIdx</i>	Adapter number (zero based)
-------------------	-----------------------------

<i>osIndex</i>	Buffer for adapter index as defined by the OS
<i>plp</i>	Buffer for adapter IP in NBO
<i>ip6</i>	Buffer for adapter IPv6 in NBO
<i>pSubnet</i>	Buffer for subnet address in NBO
<i>pBcast</i>	Buffer for bcast address in NBO
<i>pWins</i>	Buffer for semicolon delimited list of WINS servers
<i>pDns</i>	Buffer for semicolon delimited list of DNS servers

Returns

NQ_FAIL when there is no adapter with the given index, NQ_SUCCESS when adapter information available

3.7 Sockets

Macros

- #define `SY_INTERNALSOCKETPOOL`
- #define `SYSocketHandle` `NQ_INT`
- #define `SYSocketSet` `fd_set`
- #define `syIsValidSocket`(`_sock`) (`_sock != ERROR`)
- #define `syInvalidSocket`() (`ERROR`)
- #define `syAddSocketToSet`(`_sock`, `_set`) `FD_SET((_sock), (_set))`
- #define `syIsSocketSet`(`_sock`, `_set`) `FD_ISSET((_sock), (_set))`
- #define `syClearSocketSet`(`_set`) `FD_ZERO((_set))`
- #define `syClearSocketFromSet`(`_sock`, `_set`) `FD_CLR((_sock), (_set))`
- #define `sySetDatagramSocketOptions`(`_sock`)
- #define `sySetStreamSocketOptions`(`_sock`)

Functions

- `NQ_BOOL syIsSocketAlive` (`SYSocketHandle` `sock`)
- `NQ_STATUS syShutdownSocket` (`SYSocketHandle` `sock`)
- `NQ_STATUS syCloseSocket` (`SYSocketHandle` `sock`)
- `NQ_STATUS syListenSocket` (`SYSocketHandle` `sock`, `NQ_INT` `backlog`)
- `SYSocketHandle syCreateSocket` (`NQ_BOOL` `stream`, `NQ_UINT` `family`)
- `NQ_STATUS syBindSocket` (`SYSocketHandle` `sock`, `const NQ_IPADDRESS *ip`, `NQ_PORT` `port`, `NQ_BOOL` `reuseAddress`)
- `NQ_STATUS syAllowBroadcastsSocket` (`SYSocketHandle` `sock`)
- `void sySetClientSocketOptions` (`SYSocketHandle` `sock`)
- `void syGetSocketPortAndIP` (`SYSocketHandle` `sock`, `NQ_IPADDRESS *ip`, `NQ_PORT *port`)
- `NQ_INT sySendToSocket` (`SYSocketHandle` `sock`, `const NQ_BYTE *buf`, `NQ_COUNT` `len`, `const NQ_IPADDRESS *ip`, `NQ_PORT` `port`)
- `NQ_STATUS syConnectSocket` (`SYSocketHandle` `sock`, `const NQ_IPADDRESS *ip`, `NQ_PORT` `port`)
- `NQ_INT sySendSocket` (`SYSocketHandle` `sock`, `const NQ_BYTE *buf`, `NQ_COUNT` `len`)
- `NQ_STATUS sySendSocketAsync` (`SYSocketHandle` `sock`, `const NQ_BYTE *buf`, `NQ_COUNT` `len`, `void(*releaseFunc)(const NQ_BYTE *)`)
- `NQ_INT sySelectSocket` (`SYSocketSet *pset`, `NQ_UINT32` `timeout`)
- `NQ_INT syRecvFromSocket` (`SYSocketHandle` `sock`, `NQ_BYTE *buf`, `NQ_COUNT` `len`, `NQ_IPADDRESS *ip`, `NQ_PORT *port`)
- `NQ_INT syRecvSocket` (`SYSocketHandle` `sock`, `NQ_BYTE *buf`, `NQ_COUNT` `len`)
- `NQ_INT syRecvSocketWithTimeout` (`SYSocketHandle` `sock`, `NQ_BYTE *buf`, `unsigned int` `len`, `unsigned int` `secs`)
- `SYSocketHandle syAcceptSocket` (`SYSocketHandle` `sock`, `NQ_IPADDRESS *ip`, `NQ_PORT *port`)
- `NQ_STATUS sySendMulticast` (`SYSocketHandle` `socket`, `const NQ_BYTE *buffer`, `NQ_COUNT` `length`, `const NQ_IPADDRESS *ip`, `NQ_PORT` `port`)
- `void sySubscribeToMulticast` (`SYSocketHandle` `socket`, `const NQ_IPADDRESS *ip`)

3.7.1 Detailed Description

Sockets

Most socket operations are BSD 4.x standard calls. However, a few very specific operations are OS-dependent. For a BSD-compliant system use definitions below

Definition of "loopback address". This may be different for different OS. The standard value is 127.0.0.1 for IPv4 and ::1 for IPv6 (in NBO), however, some OS require a value of 0.

3.7.2 Macro Definition Documentation

3.7.2.1 #define SY_INTERNALSOCKETPOOL

Define this parameter to use internal socket pool, comment for per-task pool

3.7.2.2 #define SYSocketHandle NQ_INT

3.7.2.3 #define SYSocketSet fd_set

3.7.2.4 #define syIsValidSocket(_sock) (_sock != ERROR)

3.7.2.5 #define syInvalidSocket() (ERROR)

3.7.2.6 #define syAddSocketToSet(_sock, _set) FD_SET((_sock), (_set))

3.7.2.7 #define syIsSocketSet(_sock, _set) FD_ISSET((_sock), (_set))

3.7.2.8 #define syClearSocketSet(_set) FD_ZERO((_set))

3.7.2.9 #define syClearSocketFromSet(_sock, _set) FD_CLR((_sock), (_set))

Remove a socket from a socket set.

Parameters

<i>_sock</i>	The socket to remove.
<i>_set</i>	: Socket set.

3.7.2.10 #define sySetDatagramSocketOptions(_sock)

3.7.2.11 #define sySetStreamSocketOptions(_sock)

3.7.3 Function Documentation

3.7.3.1 NQ_BOOL syIsSocketAlive (SYSocketHandle sock)

Detecting whether a socket is still alive

Parameters

<i>sock</i>	Socket id
-------------	-----------

Returns

This method is said to work on any BSD socket system: issue select() with a zero timeout. on dead socket this should return error instead of zero

3.7.3.2 NQ_STATUS syShutdownSocket (SYSocketHandle sock)

Stop socket operations and disconnect the socket if it was connected

Parameters

<i>sock</i>	Socket id
-------------	-----------

Returns

NQ_SUCCESS or NQ_FAIL

Note

This method is said to work on any BSD socket system

3.7.3.3 NQ_STATUS syCloseSocket (SYSocketHandle *sock*)

Close socket

Parameters

<i>sock</i>	Socket id
-------------	-----------

Returns

NQ_SUCCESS or NQ_FAIL

3.7.3.4 NQ_STATUS syListenSocket (SYSocketHandle *sock*, NQ_INT *backlog*)

Listen on server socket

Parameters

<i>sock</i>	Socket id
<i>backlog</i>	Max number of requests in queue

Returns

NQ_SUCCESS or NQ_FAIL

3.7.3.5 SYSocketHandle syCreateSocket (NQ_BOOL *stream*, NQ_UINT *family*)

Create new socket

Parameters

<i>stream</i>	TRUE for TCP socket, FALSE for UDP socket
<i>family</i>	CM_IPADDR_IPV4 for IPv4, CM_IPADDR_IPV6 for IPv6

Returns

New socket or invalid socket handle

3.7.3.6 NQ_STATUS syBindSocket (SYSocketHandle *sock*, const NQ_IPADDRESS **ip*, NQ_PORT *port*, NQ_BOOL *reuseAddress*)

Bind socket to IP and port

Parameters

<i>sock</i>	Socket handle
<i>ip</i>	IP to bind to in NBO
<i>port</i>	Port to bind to in NBO
<i>reuseAddress</i>	Whether to reuse address

Returns

NQ_SUCCESS or NQ_FAIL

3.7.3.7 NQ_STATUS syAllowBroadcastsSocket (SYSocketHandle *sock*)

Allow broadcasts on an UDP socket

Parameters

<i>sock</i>	Socket handle
-------------	---------------

Returns

NQ_SUCCESS or NQ_FAIL

3.7.3.8 void sySetClientSocketOptions (SYSocketHandle *sock*)

Tune new client socket

Parameters

<i>sock</i>	Socket handle
-------------	---------------

3.7.3.9 void syGetSocketPortAndIP (SYSocketHandle *sock*, NQ_IPADDRESS * *ip*, NQ_PORT * *port*)

Get IP and port the socket is bound to

Parameters

<i>sock</i>	Socket handle
<i>ip</i>	Buffer for IP address in NBO
<i>port</i>	Buffer for port number in NBO

3.7.3.10 NQ_INT sySendToSocket (SYSocketHandle *sock*, const NQ_BYTE * *buf*, NQ_COUNT *len*, const NQ_IPADDRESS * *ip*, NQ_PORT *port*)

Send a UDP message to a specific addressee

Parameters

<i>sock</i>	Socket handle
<i>buf</i>	Data to send

<i>len</i>	Number of bytes to send
<i>ip</i>	Number of bytes to send
<i>port</i>	Port number to send to in NBO

Returns

Number of bytes sent or NQ_FAIL

3.7.3.11 NQ_STATUS syConnectSocket (SYSocketHandle sock, const NQ_IPADDRESS * ip, NQ_PORT port)

Connect to a remote server port

Parameters

<i>sock</i>	Socket handle
<i>ip</i>	IP address of the server in NBO
<i>port</i>	Port number of the server in NBO

Returns

NQ_SUCCESS or NQ_FAIL

3.7.3.12 NQ_INT sySendSocket (SYSocketHandle sock, const NQ_BYTE * buf, NQ_COUNT len)

Send bytes over a connected socket

Parameters

<i>sock</i>	Socket handle
<i>buf</i>	Data to send
<i>len</i>	Number of bytes to send

Returns

NQ_SUCCESS or NQ_FAIL

3.7.3.13 NQ_STATUS sySendSocketAsync (SYSocketHandle sock, const NQ_BYTE * buf, NQ_COUNT len, void(*) (const NQ_BYTE *) releaseFunc)

Send bytes asynchronously over a connected socket

Parameters

<i>sock</i>	Socket handle
<i>buf</i>	Data to send
<i>len</i>	Number of bytes to send
<i>releaseFunc</i>	Callback function for releasing the buffer

Returns

NQ_SUCCESS or NQ_FAIL

3.7.3.14 NQ_INT sySelectSocket (SYSocketSet * *pset*, NQ_UINT32 *timeout*)

Select on sockets

Parameters

<i>pset</i>	Pointer to the file set
<i>timeout</i>	Timeout in seconds

Returns

Number of sockets with data pending, zero on timeout or NQ_FAIL on error

3.7.3.15 NQ_INT syRecvFromSocket (SYSocketHandle *sock*, NQ_BYTE * *buf*, NQ_COUNT *len*, NQ_IPADDRESS * *ip*, NQ_PORT * *port*)

Receive a UDP message

Parameters

<i>sock</i>	Socket handle
<i>buf</i>	Receive buffer
<i>len</i>	Buffer length
<i>ip</i>	Buffer for sender IP address in NBO
<i>port</i>	Buffer for sender port in NBO

Returns

Number of bytes received or NQ_FAIL

3.7.3.16 NQ_INT syRecvSocket (SYSocketHandle *sock*, NQ_BYTE * *buf*, NQ_COUNT *len*)

Receive a UDP message from any sender

Parameters

<i>sock</i>	Socket handle
<i>buf</i>	Receive buffer
<i>len</i>	Buffer length

Returns

Number of bytes received or NQ_FAIL

3.7.3.17 NQ_INT syRecvSocketWithTimeout (SYSocketHandle *sock*, NQ_BYTE * *buf*, unsigned int *len*, unsigned int *secs*)

Receive from a datagram or a TCP stream or time out if no data on sockets.

Parameters

<i>sock</i>	Socket handle.
<i>buf</i>	Receive buffer.
<i>len</i>	Buffer size.

<i>secs</i>	Number of seconds to wait for data on sockets.
-------------	--

Returns

Number of bytes received or *NQ_FAIL* on error.

3.7.3.18 **SYSocketHandle** **syAcceptSocket** (**SYSocketHandle** *sock*, **NQ_IPADDRESS** * *ip*, **NQ_PORT** * *port*)

Accept client socket

Parameters

<i>sock</i>	Server socket handle
<i>ip</i>	Buffer for client IP address in NBO
<i>port</i>	Buffer for client port in NBO

Returns

New socket ID or invalid handle

3.7.3.19 **NQ_STATUS** **sySendMulticast** (**SYSocketHandle** *socket*, **const NQ_BYTE** * *buffer*, **NQ_COUNT** *length*, **const NQ_IPADDRESS** * *ip*, **NQ_PORT** *port*)

Send multicast datagram

Parameters

<i>socket</i>	Socket handle
<i>buffer</i>	Data to send
<i>length</i>	Number of bytes to send
<i>ip</i>	Destination IP
<i>port</i>	Destination port

Returns

NQ_FAIL or number of bytes sent

3.7.3.20 **void** **sySubscribeToMulticast** (**SYSocketHandle** *socket*, **const NQ_IPADDRESS** * *ip*)

Subscribe IP address to remote socket as a multicast

Parameters

<i>socket</i>	Remote socket handle
<i>ip</i>	Pointer to the IP address to subscribe. This value should be a multicast address in Network Byte Order (NBO).

Note

This function should analyze the IP address type (either IPv4 or IPv6) and builds IP address in the system format (e.g., -sockaddr_in/sockaddr_in6) accordingly before subscribe it.

3.8 Tasks

Macros

- #define `syGetPid()` `getpid()`

3.8.1 Detailed Description

Tasks

Task management calls. We assume that the target system answers the following generic model:

- a task has a unique id (PID) that may be mapped onto a unique 32 bit number

3.8.2 Macro Definition Documentation

3.8.2.1 #define `syGetPid()` `getpid()`

3.9 Directories

Macros

- #define SYDirectory DIR*
- #define syInvalidateDirectory(_pd) *(_pd) = NULL
- #define syIsValidDirectory(_d) (_d != NULL)

Functions

- NQ_STATUS syCreateDirectory (const NQ_WCHAR *name)
- NQ_STATUS syDeleteDirectory (const NQ_WCHAR *name)
- SYDirectory syOpenDirectory (const NQ_WCHAR *name)
- NQ_STATUS syFirstDirectoryFile (const NQ_WCHAR *name, SYDirectory *pDir, const NQ_WCHAR **fileName)
- NQ_STATUS syNextDirectoryFile (SYDirectory dir, const NQ_WCHAR **fileName)
- NQ_STATUS syCloseDirectory (SYDirectory dir)

3.9.1 Detailed Description

3.9.2 Macro Definition Documentation

3.9.2.1 #define SYDirectory DIR*

3.9.2.2 #define syInvalidateDirectory(_pd) *(_pd) = NULL

3.9.2.3 #define syIsValidDirectory(_d) (_d != NULL)

3.9.3 Function Documentation

3.9.3.1 NQ_STATUS syCreateDirectory (const NQ_WCHAR * name)

Create directory

Parameters

<i>name</i>	Full directory path
-------------	---------------------

Returns

NQ_SUCCESS or NQ_FAIL

3.9.3.2 NQ_STATUS syDeleteDirectory (const NQ_WCHAR * name)

Delete directory

Parameters

<i>name</i>	Full directory path
-------------	---------------------

Returns

NQ_SUCCESS or NQ_FAIL

3.9.3.3 SYDirectory syOpenDirectory (const NQ_WCHAR * *name*)

Open directory by name

Parameters

<i>name</i>	Full directory path
-------------	---------------------

Returns

Directory handle or invalid handle

3.9.3.4 NQ_STATUS syFirstDirectoryFile (const NQ_WCHAR * *name*, SYDirectory * *pDir*, const NQ_WCHAR ** *fileName*)

Open directory and read the first entry

Parameters

<i>name</i>	Full directory path
<i>pDir</i>	Buffer for directory handle
<i>fileName</i>	Buffer for a pointer to the file name

Returns

NQ_SUCCESS or NQ_FAIL

3.9.3.5 NQ_STATUS syNextDirectoryFile (SYDirectory *dir*, const NQ_WCHAR ** *fileName*)

Read next directory entry

Parameters

<i>dir</i>	Directory handle
<i>fileName</i>	Buffer for a pointer to the file name

Returns

NQ_SUCCESS or NQ_FAIL

3.9.3.6 NQ_STATUS syCloseDirectory (SYDirectory *dir*)

Close directory handle

Parameters

<i>dir</i>	Directory handle
------------	------------------

Returns

NQ_SUCCESS if the operation succeeded, NQ_FAIL if an error occurred.

3.10 Files

Macros

- #define SY_EXTENDFILENOTSUPPORTED
- #define SY_PATHSEPARATOR '/'
- #define SYFile int
- #define syInvalidateFile(_f) *_f = ERROR
- #define syIsValidFile(_file) (_file!=ERROR)
- #define syInvalidFile() (ERROR)
- #define SY_CP_FIRSTILLEGALCHAR {0xe5}
- #define SY_CP_ANYILLEGALCHAR {0x7c, 0x5c}
- #define syFlushFile(_file) ((fsync(_file)==OK)? NQ_SUCCESS:NQ_FAIL)
- #define syGetSecurityDescriptor(_f, _i, _b) udGetSecurityDescriptor(_f, _i, _b)
- #define sySetSecurityDescriptor(_f, _i, _b, _l) udSetSecurityDescriptor(_f, _i, _b, _l)

Functions

- SYFile syCreateFile (const NQ_WCHAR *name, NQ_BOOL denyread, NQ_BOOL denyexecute, NQ_BOOL denywrite)
- NQ_STATUS syDeleteFile (const NQ_WCHAR *name)
- NQ_STATUS syRenameFile (const NQ_WCHAR *oldName, const NQ_WCHAR *newName)
- SYFile syOpenFileForRead (const NQ_WCHAR *name, NQ_BOOL denyread, NQ_BOOL denyexecute, NQ_BOOL denywrite)
- SYFile syOpenFileForWrite (const NQ_WCHAR *name, NQ_BOOL denyread, NQ_BOOL denyexecute, NQ_BOOL denywrite)
- SYFile syOpenFileForReadWrite (const NQ_WCHAR *name, NQ_BOOL denyread, NQ_BOOL denyexecute, NQ_BOOL denywrite)
- NQ_STATUS syTruncateFile (SYFile file, NQ_UINT32 offLow, NQ_UINT32 offHigh)
- NQ_INT syReadFile (SYFile file, NQ_BYTE *buf, NQ_COUNT len)
- NQ_INT syWriteFile (SYFile file, const NQ_BYTE *buf, NQ_COUNT len)
- NQ_STATUS syCloseFile (SYFile fd)
- NQ_UINT32 sySeekFileCurrent (SYFile file, NQ_INT32 off, NQ_INT32 offHigh)
- NQ_UINT32 sySeekFileStart (SYFile file, NQ_UINT32 off, NQ_UINT32 offHigh)
- NQ_UINT32 sySeekFileEnd (SYFile file, NQ_INT32 off, NQ_INT32 offHigh)
- NQ_STATUS syGetFileInformation (SYFile file, const NQ_WCHAR *fileName, SYFileInformation *fileInfo)
- NQ_STATUS syGetFileInformationByName (const NQ_WCHAR *fileName, SYFileInformation *fileInfo)
- NQ_STATUS syGetFileSize (SYFile file, NQ_UINT64 *size)
- NQ_STATUS sySetFileInformation (const NQ_WCHAR *fileName, SYFile handle, const SYFileInformation *fileInfo)
- NQ_STATUS syGetVolumeInformation (const NQ_WCHAR *name, SYVolumeInformation *info)

3.10.1 Detailed Description

3.10.2 Macro Definition Documentation

3.10.2.1 #define SY_EXTENDFILENOTSUPPORTED

On some platforms ftruncate() might not support extending file. In this case open this define to simulate extending files

3.10.2.2 #define SY_PATHSEPARATOR '/'

Path separator character

3.10.2.3 #define SYFile int

File handle

3.10.2.4 #define syInvalidateFile(_f) *_f = ERROR

Set invalid file handle

3.10.2.5 #define syIsValidFile(_file) (_file!=ERROR)

Check file handle

3.10.2.6 #define syInvalidFile() (ERROR)

3.10.2.7 #define SY_CP_FIRSTILLEGALCHAR {0xe5}

Characters which are not acceptable by the file system as a file name

3.10.2.8 #define SY_CP_ANYILLEGALCHAR {0x7c, 0x5c}

Characters which are not acceptable by the file system as a file name

3.10.2.9 #define syFlushFile(_file) ((fsync(_file)==OK)? NQ_SUCCESS:NQ_FAIL)

3.10.2.10 #define syGetSecurityDescriptor(_f, _i, _b) udGetSecurityDescriptor(_f, _i, _b)

This function is actually user-defined. These macros are used for properly redefining file descriptors

3.10.2.11 #define sySetSecurityDescriptor(_f, _i, _b, _l) udSetSecurityDescriptor(_f, _i, _b, _l)

This function is actually user-defined. These macros are used for properly redefining file descriptors

3.10.3 Function Documentation

3.10.3.1 SYFile syCreateFile (const NQ_WCHAR * name, NQ_BOOL denyread, NQ_BOOL denyexecute, NQ_BOOL denywrite)

Create and open new file

Parameters

<i>name</i>	File name
<i>denyread</i>	True - to deny sharing for read
<i>denyexecute</i>	True - to deny sharing for execute
<i>denywrite</i>	True - to deny sharing for write

Returns

File handle or invalid handle

3.10.3.2 NQ_STATUS syDeleteFile (const NQ_WCHAR * *name*)

Delete file

Parameters

<i>name</i>	File name
-------------	-----------

Returns

NQ_SUCCESS or NQ_FAIL

3.10.3.3 NQ_STATUS syRenameFile (const NQ_WCHAR * *oldName*, const NQ_WCHAR * *newName*)

Rename file

Parameters

<i>oldName</i>	File name
<i>newName</i>	New file name

Returns

NQ_SUCCESS or NQ_FAIL

3.10.3.4 SYFile syOpenFileForRead (const NQ_WCHAR * *name*, NQ_BOOL *denyread*, NQ_BOOL *denyexecute*, NQ_BOOL *denywrite*)

Open file for reading

Parameters

<i>name</i>	File name
<i>denyread</i>	True - to deny sharing for read
<i>denyexecute</i>	True - to deny sharing for execute
<i>denywrite</i>	True - to deny sharing for write

Returns

File handle or invalid handle

3.10.3.5 SYFile syOpenFileForWrite (const NQ_WCHAR * *name*, NQ_BOOL *denyread*, NQ_BOOL *denyexecute*, NQ_BOOL *denywrite*)

Open file for writing

Parameters

<i>name</i>	File name
<i>denyread</i>	True - to deny sharing for read
<i>denyexecute</i>	True - to deny sharing for execute

<i>denywrite</i>	True - to deny sharing for write
------------------	----------------------------------

Returns

File handle or invalid handle

3.10.3.6 SYFile syOpenFileForReadWrite (const NQ_WCHAR * *name*, NQ_BOOL *denyread*, NQ_BOOL *denyexecute*, NQ_BOOL *denywrite*)

Open file for reading and writing

Parameters

<i>name</i>	File name
<i>denyread</i>	True - to deny sharing for read
<i>denyexecute</i>	True - to deny sharing for execute
<i>denywrite</i>	True - to deny sharing for write

Returns

File handle or invalid handle

3.10.3.7 NQ_STATUS syTruncateFile (SYFile *file*, NQ_UINT32 *offLow*, NQ_UINT32 *offHigh*)

Truncate file

Parameters

<i>file</i>	File handle
<i>offLow</i>	Offset low
<i>offHigh</i>	Offset high

Returns

NQ_SUCCESS or NQ_FAIL

3.10.3.8 NQ_INT syReadFile (SYFile *file*, NQ_BYTE * *buf*, NQ_COUNT *len*)

Read bytes from file

Parameters

<i>file</i>	File handle
<i>buf</i>	Buffer for data
<i>len</i>	Number of bytes to read

Returns

Number of bytes read, zero on end of file, or NQ_FAIL

3.10.3.9 NQ_INT syWriteFile (SYFile *file*, const NQ_BYTE * *buf*, NQ_COUNT *len*)

Write bytes into file

Parameters

<i>file</i>	File handle
<i>buf</i>	Bytes to write
<i>len</i>	Number of bytes to write

Returns

Number of bytes written or NQ_FAIL

3.10.3.10 NQ_STATUS syCloseFile (SYFile *fd*)

Close opened file

Parameters

<i>fd</i>	Handle of the file
-----------	--------------------

Returns

NQ_SUCCESS if the file was close or NQ_FAIL on error

3.10.3.11 NQ_UINT32 sySeekFileCurrent (SYFile *file*, NQ_INT32 *off*, NQ_INT32 *offHigh*)

Position file relatively from the current position

Parameters

<i>file</i>	File handle
<i>off</i>	Low 32 bits of the offset
<i>offHigh</i>	High 32 bits of the offset

Returns

New file position or NQ_FAIL

3.10.3.12 NQ_UINT32 sySeekFileStart (SYFile *file*, NQ_UINT32 *off*, NQ_UINT32 *offHigh*)

Position file from the beginning

Parameters

<i>file</i>	File handle
<i>off</i>	Low 32 bits of the offset
<i>offHigh</i>	High 32 bits of the offset

Returns

New file position or NQ_FAIL

3.10.3.13 NQ_UINT32 sySeekFileEnd (SYFile *file*, NQ_INT32 *off*, NQ_INT32 *offHigh*)

Position file from the end

Parameters

<i>file</i>	File handle
<i>off</i>	Low 32 bits of the offset
<i>offHigh</i>	High 32 bits of the offset

Returns

New file position or NQ_FAIL

3.10.3.14 NQ_STATUS syGetFileInformation (SYFile *file*, const NQ_WCHAR * *fileName*, SYFileInformation * *fileInfo*)

Read file information structure by file handle

Parameters

<i>file</i>	File id
<i>fileName</i>	File name
<i>fileInfo</i>	File information structure

Returns

NQ_SUCCESS or NQ_FAIL

3.10.3.15 NQ_STATUS syGetFileInformationByName (const NQ_WCHAR * *fileName*, SYFileInformation * *fileInfo*)

Read file information structure by file name

Parameters

<i>fileName</i>	File name
<i>fileInfo</i>	File information structure

Returns

NQ_SUCCESS or NQ_FAIL

3.10.3.16 NQ_STATUS syGetFileSize (SYFile *file*, NQ_UINT64 * *size*)

Read file size by file handle

Parameters

<i>file</i>	File id
<i>size</i>	File size

Returns

NQ_SUCCESS or NQ_FAIL

3.10.3.17 NQ_STATUS sySetFileInformation (const NQ_WCHAR * *fileName*, SYFile *handle*, const SYFileInformation * *fileInfo*)

Update file information by either file name or file handle

Parameters

<i>fileName</i>	File name
<i>handle</i>	File handle
<i>fileInfo</i>	File information structure

Returns

NQ_SUCCESS or NQ_FAIL

**3.10.3.18 NQ_STATUS syGetVolumeInformation (const NQ_WCHAR * *name*,
SYVolumeInformation * *info*)**

Query volume information

Parameters

<i>name</i>	Volume name
<i>info</i>	Buffer for information

Returns

NQ_SUCCESS or NQ_FAIL

3.11 Input Output

Macros

- #define `syPrintf` printf
- #define `syFprintf` fprintf
- #define `sySprintf` sprintf
- #define `sySnprintf` snprintf
- #define `syVsnprintf` vsnprintf
- #define `sySscanf` sscanf
- #define `syScanf` scanf
- #define `syGetchar` getchar
- #define `syFlush` fflush

3.11.1 Detailed Description

3.11.2 Macro Definition Documentation

3.11.2.1 #define syPrintf printf

3.11.2.2 #define syFprintf fprintf

3.11.2.3 #define sySprintf sprintf

3.11.2.4 #define sySnprintf snprintf

3.11.2.5 #define syVsnprintf vsnprintf

3.11.2.6 #define sySscanf sscanf

3.11.2.7 #define syScanf scanf

3.11.2.8 #define syGetchar getchar

3.11.2.9 #define syFlush fflush

3.12 File Locking

Macros

- #define `syUnlockFile`(`_file`, `_offsetHigh`, `_offsetLow`, `_lengthHigh`, `_lengthLow`, `_timeout`) ((`_file`==ERROR)? NQ_FAIL:NQ_SUCCESS)
- #define `syLockFile`(`_file`, `_offsetHigh`, `_offsetLow`, `_lengthHigh`, `_lengthLow`, `_lockType`, `_oplockLevel`) ((`_file`==ERROR)? NQ_FAIL:NQ_SUCCESS)

3.12.1 Detailed Description

File locking

The default implementation is mere a placeholder

3.12.2 Macro Definition Documentation

3.12.2.1 #define `syUnlockFile`(`_file`, `_offsetHigh`, `_offsetLow`, `_lengthHigh`, `_lengthLow`, `_timeout`) ((`_file`==ERROR)? NQ_FAIL:NQ_SUCCESS)

3.12.2.2 #define `syLockFile`(`_file`, `_offsetHigh`, `_offsetLow`, `_lengthHigh`, `_lengthLow`, `_lockType`, `_oplockLevel`) ((`_file`==ERROR)? NQ_FAIL:NQ_SUCCESS)

3.13 Direct Transfer

Functions

- NQ_STATUS [syDtStartPacket](#) (SYSocketHandle sock)
- void [syDtEndPacket](#) (SYSocketHandle sock)
- NQ_STATUS [syDtFromSocket](#) (SYSocketHandle sock, SYFile file, NQ_COUNT *len)
- NQ_STATUS [syDtToSocket](#) (SYSocketHandle sock, SYFile file, NQ_COUNT *len)

3.13.1 Detailed Description

3.13.2 Function Documentation

3.13.2.1 NQ_STATUS syDtStartPacket (SYSocketHandle sock)

Start fragmented packet

Parameters

<i>sock</i>	Socket handle
-------------	---------------

Returns

NQ_FAIL when this operation is not available NQ_SUCCESS when operation succeeded

3.13.2.2 void syDtEndPacket (SYSocketHandle sock)

End fragmented packet

Parameters

<i>sock</i>	Socket handle
-------------	---------------

3.13.2.3 NQ_STATUS syDtFromSocket (SYSocketHandle sock, SYFile file, NQ_COUNT * len)

Transfer bytes from socket to file

Parameters

<i>sock</i>	Socket handle
<i>file</i>	File handle
<i>len</i>	Number of bytes to transfer, OUT bytes transferred

Returns

NQ_FAIL on error or NQ_SUCCESS when operation succeeded

3.13.2.4 NQ_STATUS syDtToSocket (SYSocketHandle sock, SYFile file, NQ_COUNT * len)

Transfer bytes from file to socket

Parameters

<i>sock</i>	Socket handle
<i>file</i>	File handle
<i>len</i>	Number of bytes to transfer, OUT bytes transferred

Returns

NQ_FAIL on error or NQ_SUCCESS when operation succeeded

3.14 Encoding

Macros

- #define [SY_UNICODEFILESYSTEM](#)

Functions

- void [syUnicodeToUTF8N](#) (NQ_CHAR *outStr, NQ_UINT outLength, const NQ_WCHAR *inWStr, NQ_UINT inLength)
- void [syUTF8ToUnicodeN](#) (NQ_WCHAR *outWStr, NQ_UINT outLength, const NQ_CHAR *inStr, NQ_UINT inLength)
- NQ_BOOL [initCodePageUTF8](#) (void)

3.14.1 Detailed Description

3.14.2 Macro Definition Documentation

3.14.2.1 #define SY_UNICODEFILESYSTEM

Whether the filesystem supports Unicode. Otherwise all filenames will be converted to ANSI even if CIFS is supporting UNICODE

3.14.3 Function Documentation

3.14.3.1 void [syUnicodeToUTF8N](#) (NQ_CHAR * *outStr*, NQ_UINT *outLength*, const NQ_WCHAR * *inWStr*, NQ_UINT *inLength*)

Convert Unicode string to UTF8

Parameters

<i>outStr</i>	Pointer to the result string in UTF8
<i>outLength</i>	Length of the result buffer
<i>inWStr</i>	Pointer to string in UNICODE to be converted
<i>inLength</i>	Length of string to be converted

3.14.3.2 void [syUTF8ToUnicodeN](#) (NQ_WCHAR * *outWStr*, NQ_UINT *outLength*, const NQ_CHAR * *inStr*, NQ_UINT *inLength*)

Convert UTF8 string to Unicode

Parameters

<i>outWStr</i>	Pointer to the result string in UNICODE
<i>outLength</i>	Length of the result buffer
<i>inStr</i>	Pointer to input string in UTF8
<i>inLength</i>	Length of string to be converted

3.14.3.3 NQ_BOOL [initCodePageUTF8](#) (void)

Check whether conversion is available on the platform

Returns

TRUE on success, *FALSE* on error.

3.15 Utils

Macros

- #define [sySetRand\(\)](#) srand((unsigned int)time(0))
- #define [syRand\(\)](#) rand()

Functions

- int [syUnixMode2DosAttr](#) (int mode)
- NQ_UINT32 [syGetLastSmbError](#) (void)
- void [sySetLastNqError](#) (NQ_STATUS nqErr)

3.15.1 Detailed Description

3.15.2 Macro Definition Documentation

3.15.2.1 #define sySetRand() srand((unsigned int)time(0))

Take seed from the system time

3.15.2.2 #define syRand() rand()

Next (pseudo)random value

3.15.3 Function Documentation

3.15.3.1 int syUnixMode2DosAttr (int *mode*)

Convert UNIX file permissions to DOS file attributes

Parameters

<i>mode</i>	UNIX file permissions
-------------	-----------------------

Returns

Matching DOS file attributes

Note

- DOS read only is represented in UNIX by removing everyone's write bit.
- DOS archive is represented in UNIX by the user's execute bit.
- DOS system is represented in UNIX by the group's execute bit.
- DOS hidden is represented in UNIX by the other's execute bit.
- DOS directory is represented in UNIX by UNIX 's directory bit.

3.15.3.2 NQ_UINT32 syGetLastSmbError (void)

Convert last system error to SMB error

Returns

SMB error

3.15.3.3 void sySetLastNqError (NQ_STATUS *nqErr*)

Convert NQ error into system error

Parameters

<i>nqErr</i>	NQ error
--------------	----------

Chapter 4

File Documentation

4.1 syopsyst.h File Reference

```
#include "sycommon.h"
```

Macros

- `#define MUTEX_DEBUG`
- `#define SY_OSNAME "RedHat Linux"`
- `#define syAssert(_stat_) assert(_stat_)`
- `#define syGetTimeInSec() time(0)`
- `#define sySleep(_secs_) sleep(_secs_)`
- `#define syUSleep(_msecs_) usleep(_msecs_)`
- `#define SYThread pthread_t`
- `#define syIsValidThread(_taskId_) TRUE`
- `#define syThreadGetCurrent pthread_self`
- `#define syThreadDestroy(_taskId_) pthread_cancel(_taskId_);`
- `#define SYMutex pthread_mutex_t`
- `#define syMutexTake(_m) syMutexTakeDebug(_m, SY_LOG_FILE, SY_LOG_LINE)`
- `#define SY_SEMAPHORE_AVAILABLE`
- `#define SYSemaphore sem_t`
- `#define sySemaphoreDelete(_s) sem_destroy(&_s)`
- `#define sySemaphoreTake(_s) sem_wait(&_s)`
- `#define sySemaphoreGive(_s) sem_post(&_s)`
- `#define SY_LOCALHOSTIP4 /*{0, 0} */ {0x7f00, 0x0001}`
- `#define SY_LOCALHOSTIP6 {0, 0, 0, 0, 0, 0, 0, 1} /* {0, 0, 0, 0, 0, 0, 0, 0} */`
- `#define SY_LINKLOCALIP 0xfe80`
- `#define SY_ANYIP4 {0, 0}`
- `#define SY_ANYIP6 {0, 0, 0, 0, 0, 0, 0, 0}`
- `#define SY_ZEROIP {0, 0, 0, 0, 0, 0, 0, 0}`
- `#define SY_ZEROIP4 0L`
- `#define SY_INTERNALSOCKETPOOL`
- `#define SYSocketHandle NQ_INT`
- `#define SYSocketSet fd_set`
- `#define syIsValidSocket(_sock) (_sock != ERROR)`
- `#define syInvalidSocket() (ERROR)`
- `#define syAddSocketToSet(_sock, _set) FD_SET((_sock), (_set))`
- `#define syIsSocketSet(_sock, _set) FD_ISSET((_sock), (_set))`
- `#define syClearSocketSet(_set) FD_ZERO((_set))`
- `#define syClearSocketFromSet(_sock, _set) FD_CLR((_sock), (_set))`
- `#define sySetDatagramSocketOptions(_sock)`

- #define [sySetStreamSocketOptions](#)(_sock)
- #define [syGetPid](#)() getpid()
- #define [SYDirectory](#) DIR*
- #define [syInvalidateDirectory](#)(_pd) *(_pd) = NULL
- #define [syIsValidDirectory](#)(_d) (_d != NULL)
- #define [SY_EXTENDFILENOTSUPPORTED](#)
- #define [SY_PATHSEPARATOR](#) '/'
- #define [SYFile](#) int
- #define [syInvalidateFile](#)(_f) *_f = ERROR
- #define [syIsValidFile](#)(_file) (_file != ERROR)
- #define [syInvalidFile](#)() (ERROR)
- #define [SY_CP_FIRSTILLEGALCHAR](#) {0xe5}
- #define [SY_CP_ANYILLEGALCHAR](#) {0x7c, 0x5c}
- #define [syFlushFile](#)(_file) ((fsync(_file)==OK)? NQ_SUCCESS:NQ_FAIL)
- #define [syGetSecurityDescriptor](#)(_f, _i, _b) udGetSecurityDescriptor(_f, _i, _b)
- #define [sySetSecurityDescriptor](#)(_f, _i, _b, _l) udSetSecurityDescriptor(_f, _i, _b, _l)
- #define [syPrintf](#) printf
- #define [syFprintf](#) fprintf
- #define [sySprintf](#) sprintf
- #define [sySnprintf](#) snprintf
- #define [syVsnprintf](#) vsnprintf
- #define [sySscanf](#) sscanf
- #define [syScanf](#) scanf
- #define [syGetchar](#) getchar
- #define [syFlush](#) fflush
- #define [syUnlockFile](#)(_file, _offsetHigh, _offsetLow, _lengthHigh, _lengthLow, _timeout) ((_file==ERROR)? NQ_FAIL:NQ_SUCCESS)
- #define [syLockFile](#)(_file, _offsetHigh, _offsetLow, _lengthHigh, _lengthLow, _lockType, _oplockLevel) ((_file==ERROR)? NQ_FAIL:NQ_SUCCESS)
- #define [syGetHostName](#)(_name, _nameLen) gethostname((_name), (_nameLen))
- #define [SY_UNICODEFILESYSTEM](#)
- #define [sySetRand](#)() srand((unsigned int)time(0))
- #define [syRand](#)() rand()

Functions

- NQ_TIME [syGetTimeInMsec](#) (void)
- NQ_TIME [syConvertTimeSpecToTimeInMsec](#) (void *val)
- NQ_UINT32 [syConvertTimeInMsecToSec](#) (NQ_TIME *timeMsec)
- NQ_INT [syGetTimeZone](#) (void)
- void [syDecomposeTime](#) (NQ_UINT32 time, SYTimeFragments *decomposed)
- NQ_UINT32 [syComposeTime](#) (const SYTimeFragments *decomposed)
- NQ_BOOL [syGmtToString](#) (NQ_BYTE *strTime, NQ_COUNT size, NQ_UINT32 t, const NQ_CHAR *fmt)
- void [syThreadStart](#) (NQ_BOOL isRT, NQ_INT priorityLevel, [SYThread](#) *taskIdPtr, void(*startpoint)(void), NQ_BOOL background)
- void [syMutexCreate](#) (SYMutex *_m)
- void [syMutexDelete](#) (SYMutex *_m)
- void [syMutexTakeDebug](#) (SYMutex *_m, const NQ_CHAR *text, const NQ_UINT line)
- void [syMutexGive](#) (SYMutex *_m)
- NQ_STATUS [sySemaphoreCreate](#) (SYSemaphore *semId, NQ_UINT count)
- void [sySemaphoreResetCounter](#) (SYSemaphore *pSemId)
- NQ_INT [sySemaphoreTimedTake](#) (SYSemaphore *sem, NQ_INT timeout)
- NQ_UINT32 [syGetIPv6ScopeId](#) (const NQ_IPADDRESS6 ip)
- NQ_BOOL [syIsSocketAlive](#) (SYSocketHandle sock)

- NQ_STATUS [syShutdownSocket](#) (SYSocketHandle sock)
- NQ_STATUS [syCloseSocket](#) (SYSocketHandle sock)
- NQ_STATUS [syListenSocket](#) (SYSocketHandle sock, NQ_INT backlog)
- [SYSocketHandle](#) [syCreateSocket](#) (NQ_BOOL stream, NQ_UINT family)
- NQ_STATUS [syBindSocket](#) (SYSocketHandle sock, const NQ_IPADDRESS *ip, NQ_PORT port, NQ_BOOL reuseAddress)
- NQ_STATUS [syAllowBroadcastsSocket](#) (SYSocketHandle sock)
- void [sySetClientSocketOptions](#) (SYSocketHandle sock)
- void [syGetSocketPortAndIP](#) (SYSocketHandle sock, NQ_IPADDRESS *ip, NQ_PORT *port)
- NQ_INT [sySendToSocket](#) (SYSocketHandle sock, const NQ_BYTE *buf, NQ_COUNT len, const NQ_IPADDRESS *ip, NQ_PORT port)
- NQ_STATUS [syConnectSocket](#) (SYSocketHandle sock, const NQ_IPADDRESS *ip, NQ_PORT port)
- NQ_INT [sySendSocket](#) (SYSocketHandle sock, const NQ_BYTE *buf, NQ_COUNT len)
- NQ_STATUS [sySendSocketAsync](#) (SYSocketHandle sock, const NQ_BYTE *buf, NQ_COUNT len, void(*releaseFunc)(const NQ_BYTE *))
- NQ_INT [sySelectSocket](#) (SYSocketSet *pset, NQ_UINT32 timeout)
- NQ_INT [syRecvFromSocket](#) (SYSocketHandle sock, NQ_BYTE *buf, NQ_COUNT len, NQ_IPADDRESS *ip, NQ_PORT *port)
- NQ_INT [syRecvSocket](#) (SYSocketHandle sock, NQ_BYTE *buf, NQ_COUNT len)
- NQ_INT [syRecvSocketWithTimeout](#) (SYSocketHandle sock, NQ_BYTE *buf, unsigned int len, unsigned int secs)
- [SYSocketHandle](#) [syAcceptSocket](#) (SYSocketHandle sock, NQ_IPADDRESS *ip, NQ_PORT *port)
- NQ_STATUS [sySendMulticast](#) (SYSocketHandle socket, const NQ_BYTE *buffer, NQ_COUNT length, const NQ_IPADDRESS *ip, NQ_PORT port)
- void [sySubscribeToMulticast](#) (SYSocketHandle socket, const NQ_IPADDRESS *ip)
- NQ_STATUS [syCreateDirectory](#) (const NQ_WCHAR *name)
- NQ_STATUS [syDeleteDirectory](#) (const NQ_WCHAR *name)
- [SYDirectory](#) [syOpenDirectory](#) (const NQ_WCHAR *name)
- NQ_STATUS [syFirstDirectoryFile](#) (const NQ_WCHAR *name, [SYDirectory](#) *pDir, const NQ_WCHAR **fileName)
- NQ_STATUS [syNextDirectoryFile](#) ([SYDirectory](#) dir, const NQ_WCHAR **fileName)
- NQ_STATUS [syCloseDirectory](#) ([SYDirectory](#) dir)
- [SYFile](#) [syCreateFile](#) (const NQ_WCHAR *name, NQ_BOOL denyread, NQ_BOOL denyexecute, NQ_BOOL denywrite)
- NQ_STATUS [syDeleteFile](#) (const NQ_WCHAR *name)
- NQ_STATUS [syRenameFile](#) (const NQ_WCHAR *oldName, const NQ_WCHAR *newName)
- [SYFile](#) [syOpenFileForRead](#) (const NQ_WCHAR *name, NQ_BOOL denyread, NQ_BOOL denyexecute, NQ_BOOL denywrite)
- [SYFile](#) [syOpenFileForWrite](#) (const NQ_WCHAR *name, NQ_BOOL denyread, NQ_BOOL denyexecute, NQ_BOOL denywrite)
- [SYFile](#) [syOpenFileForReadWrite](#) (const NQ_WCHAR *name, NQ_BOOL denyread, NQ_BOOL denyexecute, NQ_BOOL denywrite)
- NQ_STATUS [syTruncateFile](#) ([SYFile](#) file, NQ_UINT32 offLow, NQ_UINT32 offHigh)
- NQ_INT [syReadFile](#) ([SYFile](#) file, NQ_BYTE *buf, NQ_COUNT len)
- NQ_INT [syWriteFile](#) ([SYFile](#) file, const NQ_BYTE *buf, NQ_COUNT len)
- NQ_STATUS [syCloseFile](#) ([SYFile](#) fd)
- NQ_UINT32 [sySeekFileCurrent](#) ([SYFile](#) file, NQ_INT32 off, NQ_INT32 offHigh)
- NQ_UINT32 [sySeekFileStart](#) ([SYFile](#) file, NQ_UINT32 off, NQ_UINT32 offHigh)
- NQ_UINT32 [sySeekFileEnd](#) ([SYFile](#) file, NQ_INT32 off, NQ_INT32 offHigh)
- NQ_STATUS [syGetFileInformation](#) ([SYFile](#) file, const NQ_WCHAR *fileName, SYFileInformation *fileInfo)
- NQ_STATUS [syGetFileInformationByName](#) (const NQ_WCHAR *fileName, SYFileInformation *fileInfo)
- NQ_STATUS [syGetFileSize](#) ([SYFile](#) file, NQ_UINT64 *size)
- NQ_STATUS [sySetFileInformation](#) (const NQ_WCHAR *fileName, [SYFile](#) handle, const SYFileInformation *fileInfo)
- NQ_STATUS [syGetVolumeInformation](#) (const NQ_WCHAR *name, SYVolumeInformation *info)

- NQ_IPADDRESS4 [syGetHostByName](#) (const char *name)
- void [syGetDnsParams](#) (NQ_CHAR *domain, NQ_IPADDRESS *server)
- void [syGetMacAddress](#) (NQ_IPADDRESS4 ip, NQ_BYTE *macBuffer)
- NQ_STATUS [syGetAdapter](#) (NQ_INDEX adapterIdx, NQ_INDEX *osIndex, NQ_IPADDRESS4 *pIp, NQ_IPADDRESS6 *ip6, NQ_IPADDRESS4 *pSubnet, NQ_IPADDRESS4 *pBcast, NQ_WCHAR *pWins, NQ_WCHAR *pDns)
- NQ_STATUS [syDtStartPacket](#) (SYSocketHandle sock)
- void [syDtEndPacket](#) (SYSocketHandle sock)
- NQ_STATUS [syDtFromSocket](#) (SYSocketHandle sock, SYFile file, NQ_COUNT *len)
- NQ_STATUS [syDtToSocket](#) (SYSocketHandle sock, SYFile file, NQ_COUNT *len)
- void [syUnicodeToUTF8N](#) (NQ_CHAR *outStr, NQ_UINT outLength, const NQ_WCHAR *inWStr, NQ_UINT inLength)
- void [syUTF8ToUnicodeN](#) (NQ_WCHAR *outWStr, NQ_UINT outLength, const NQ_CHAR *inStr, NQ_UINT inLength)
- NQ_BOOL [initCodePageUTF8](#) (void)
- int [syUnixMode2DosAttr](#) (int mode)
- NQ_UINT32 [syGetLastSmbError](#) (void)
- void [sySetLastNqError](#) (NQ_STATUS nqErr)

4.1.1 Macro Definition Documentation

4.1.1.1 #define MUTEX_DEBUG

When defined each mutex create, take, destroy will be logged

4.1.1.2 #define SY_OSNAME "RedHat Linux"

Operating system name

4.1.1.3 #define syAssert(_stat_) assert(_stat_)

Assert macro

Index

- Direct Transfer, [37](#)
 - [syDtEndPacket, 37](#)
 - [syDtFromSocket, 37](#)
 - [syDtStartPacket, 37](#)
 - [syDtToSocket, 37](#)
- Directories, [25](#)
 - [SYDirectory, 25](#)
 - [syCloseDirectory, 27](#)
 - [syCreateDirectory, 25](#)
 - [syDeleteDirectory, 25](#)
 - [syFirstDirectoryFile, 27](#)
 - [syInvalidateDirectory, 25](#)
 - [syIsValidDirectory, 25](#)
 - [syNextDirectoryFile, 27](#)
 - [syOpenDirectory, 25](#)
- Encoding, [39](#)
 - [initCodePageUTF8, 39](#)
 - [SY_UNICODEFILESYSTEM, 39](#)
 - [syUTF8ToUnicodeN, 39](#)
 - [syUnicodeToUTF8N, 39](#)
- File Locking, [36](#)
 - [syLockFile, 36](#)
 - [syUnlockFile, 36](#)
- Files, [28](#)
 - [SY_CP_ANYILLEGALCHAR, 29](#)
 - [SY_CP_FIRSTILLEGALCHAR, 29](#)
 - [SY_EXTENDFILENOTSUPPORTED, 28](#)
 - [SY_PATHSEPARATOR, 28](#)
 - [SYFile, 28](#)
 - [syCloseFile, 32](#)
 - [syCreateFile, 29](#)
 - [syDeleteFile, 29](#)
 - [syFlushFile, 29](#)
 - [syGetFileInformation, 33](#)
 - [syGetFileInformationByName, 33](#)
 - [syGetFileSize, 33](#)
 - [syGetSecurityDescriptor, 29](#)
 - [syGetVolumeInformation, 34](#)
 - [syInvalidFile, 29](#)
 - [syInvalidateFile, 29](#)
 - [syIsValidFile, 29](#)
 - [syOpenFileForRead, 30](#)
 - [syOpenFileForReadWrite, 31](#)
 - [syOpenFileForWrite, 30](#)
 - [syReadFile, 31](#)
 - [syRenameFile, 30](#)
 - [sySeekFileCurrent, 32](#)
 - [sySeekFileEnd, 32](#)
 - [sySeekFileStart, 32](#)
 - [sySetFileInformation, 33](#)
 - [sySetSecurityDescriptor, 29](#)
 - [syTruncateFile, 31](#)
 - [syWriteFile, 31](#)
- initCodePageUTF8
 - [Encoding, 39](#)
- Input Output, [35](#)
 - [syFflush, 35](#)
 - [syFprintf, 35](#)
 - [syGetchar, 35](#)
 - [syPrintf, 35](#)
 - [syScanf, 35](#)
 - [sySnpprintf, 35](#)
 - [sySprintf, 35](#)
 - [sySscanf, 35](#)
 - [syVsnprintf, 35](#)
- MUTEX_DEBUG
 - [syopsyst.h, 46](#)
- Mutexes, [9](#)
 - [SYMutex, 9](#)
 - [syMutexCreate, 9](#)
 - [syMutexDelete, 9](#)
 - [syMutexGive, 10](#)
 - [syMutexTake, 9](#)
 - [syMutexTakeDebug, 9](#)
- Network, [13](#)
 - [SY_ANYIP4, 13](#)
 - [SY_ANYIP6, 13](#)
 - [SY_LINKLOCALIP, 13](#)
 - [SY_LOCALHOSTIP4, 13](#)
 - [SY_LOCALHOSTIP6, 13](#)
 - [SY_ZEROIP, 13](#)
 - [SY_ZEROIP4, 13](#)
 - [syGetAdapter, 14](#)
 - [syGetDnsParams, 14](#)
 - [syGetHostByName, 14](#)
 - [syGetHostName, 13](#)
 - [syGetIPv6Scopeld, 13](#)
 - [syGetMacAddress, 14](#)
- SY_ANYIP4
 - [Network, 13](#)
- SY_ANYIP6
 - [Network, 13](#)
- SY_CP_ANYILLEGALCHAR
 - [Files, 29](#)
- SY_CP_FIRSTILLEGALCHAR
 - [Files, 29](#)

SY_EXTENDFILENOTSUPPORTED
Files, [28](#)
SY_INTERNALSOCKETPOOL
Sockets, [17](#)
SY_LINKLOCALIP
Network, [13](#)
SY_LOCALHOSTIP4
Network, [13](#)
SY_LOCALHOSTIP6
Network, [13](#)
SY_OSNAME
syopsyst.h, [46](#)
SY_PATHSEPARATOR
Files, [28](#)
SY_SEMAPHORE_AVAILABLE
Semaphores, [11](#)
SY_UNICODEFILESYSTEM
Encoding, [39](#)
SY_ZEROIP
Network, [13](#)
SY_ZEROIP4
Network, [13](#)
SYDirectory
Directories, [25](#)
SYFile
Files, [28](#)
SYMutex
Mutexes, [9](#)
SYSemaphore
Semaphores, [11](#)
SYSocketHandle
Sockets, [17](#)
SYSocketSet
Sockets, [17](#)
SYThread
Threads, [8](#)
Semaphores, [11](#)
SY_SEMAPHORE_AVAILABLE, [11](#)
SYSemaphore, [11](#)
sySemaphoreCreate, [11](#)
sySemaphoreDelete, [11](#)
sySemaphoreGive, [11](#)
sySemaphoreResetCounter, [12](#)
sySemaphoreTake, [11](#)
sySemaphoreTimedTake, [12](#)
Sockets, [16](#)
SY_INTERNALSOCKETPOOL, [17](#)
SYSocketHandle, [17](#)
SYSocketSet, [17](#)
syAcceptSocket, [23](#)
syAddSocketToSet, [17](#)
syAllowBroadcastsSocket, [19](#)
syBindSocket, [18](#)
syClearSocketFromSet, [17](#)
syClearSocketSet, [17](#)
syCloseSocket, [18](#)
syConnectSocket, [20](#)
syCreateSocket, [18](#)
syGetSocketPortAndIP, [19](#)
syInvalidSocket, [17](#)
syIsSocketAlive, [17](#)
syIsSocketSet, [17](#)
syIsValidSocket, [17](#)
syListenSocket, [18](#)
syRecvFromSocket, [22](#)
syRecvSocket, [22](#)
syRecvSocketWithTimeout, [22](#)
sySelectSocket, [20](#)
sySendMulticast, [23](#)
sySendSocket, [20](#)
sySendSocketAsync, [20](#)
sySendToSocket, [19](#)
sySetClientSocketOptions, [19](#)
sySetDatagramSocketOptions, [17](#)
sySetStreamSocketOptions, [17](#)
syShutdownSocket, [17](#)
sySubscribeToMulticast, [23](#)
syAcceptSocket
Sockets, [23](#)
syAddSocketToSet
Sockets, [17](#)
syAllowBroadcastsSocket
Sockets, [19](#)
syAssert
syopsyst.h, [46](#)
syBindSocket
Sockets, [18](#)
syClearSocketFromSet
Sockets, [17](#)
syClearSocketSet
Sockets, [17](#)
syCloseDirectory
Directories, [27](#)
syCloseFile
Files, [32](#)
syCloseSocket
Sockets, [18](#)
syComposeTime
Time, [6](#)
syConnectSocket
Sockets, [20](#)
syConvertTimeInMsecToSec
Time, [6](#)
syConvertTimeSpecToTimeInMsec
Time, [4](#)
syCreateDirectory
Directories, [25](#)
syCreateFile
Files, [29](#)
syCreateSocket
Sockets, [18](#)
syDecomposeTime
Time, [6](#)
syDeleteDirectory
Directories, [25](#)
syDeleteFile

Files, [29](#)
syDtEndPacket
 Direct Transfer, [37](#)
syDtFromSocket
 Direct Transfer, [37](#)
syDtStartPacket
 Direct Transfer, [37](#)
syDtToSocket
 Direct Transfer, [37](#)
syFflush
 Input Output, [35](#)
syFirstDirectoryFile
 Directories, [27](#)
syFlushFile
 Files, [29](#)
syFprintf
 Input Output, [35](#)
syGetAdapter
 Network, [14](#)
syGetDnsParams
 Network, [14](#)
syGetFileInformation
 Files, [33](#)
syGetFileInformationByName
 Files, [33](#)
syGetFileSize
 Files, [33](#)
syGetHostByName
 Network, [14](#)
syGetHostName
 Network, [13](#)
syGetIPv6Scopeld
 Network, [13](#)
syGetLastSmbError
 Utils, [41](#)
syGetMacAddress
 Network, [14](#)
syGetPid
 Tasks, [24](#)
syGetSecurityDescriptor
 Files, [29](#)
syGetSocketPortAndIP
 Sockets, [19](#)
syGetTimeInMsec
 Time, [4](#)
syGetTimeInSec
 Time, [4](#)
syGetTimeZone
 Time, [6](#)
syGetVolumeInformation
 Files, [34](#)
syGetchar
 Input Output, [35](#)
syGmtToString
 Time, [6](#)
syInvalidFile
 Files, [29](#)
syInvalidSocket
 Sockets, [17](#)
syInvalidateDirectory
 Directories, [25](#)
syInvalidateFile
 Files, [29](#)
syIsSocketAlive
 Sockets, [17](#)
syIsSocketSet
 Sockets, [17](#)
syIsValidDirectory
 Directories, [25](#)
syIsValidFile
 Files, [29](#)
syIsValidSocket
 Sockets, [17](#)
syIsValidThread
 Threads, [8](#)
syListenSocket
 Sockets, [18](#)
syLockFile
 File Locking, [36](#)
syMutexCreate
 Mutexes, [9](#)
syMutexDelete
 Mutexes, [9](#)
syMutexGive
 Mutexes, [10](#)
syMutexTake
 Mutexes, [9](#)
syMutexTakeDebug
 Mutexes, [9](#)
syNextDirectoryFile
 Directories, [27](#)
syOpenDirectory
 Directories, [25](#)
syOpenFileForRead
 Files, [30](#)
syOpenFileForReadWrite
 Files, [31](#)
syOpenFileForWrite
 Files, [30](#)
syPrintf
 Input Output, [35](#)
syRand
 Utils, [41](#)
syReadFile
 Files, [31](#)
syRecvFromSocket
 Sockets, [22](#)
syRecvSocket
 Sockets, [22](#)
syRecvSocketWithTimeout
 Sockets, [22](#)
syRenameFile
 Files, [30](#)
syScanf
 Input Output, [35](#)
sySeekFileCurrent

- Files, [32](#)
- sySeekFileEnd
 - Files, [32](#)
- sySeekFileStart
 - Files, [32](#)
- sySelectSocket
 - Sockets, [20](#)
- sySemaphoreCreate
 - Semaphores, [11](#)
- sySemaphoreDelete
 - Semaphores, [11](#)
- sySemaphoreGive
 - Semaphores, [11](#)
- sySemaphoreResetCounter
 - Semaphores, [12](#)
- sySemaphoreTake
 - Semaphores, [11](#)
- sySemaphoreTimedTake
 - Semaphores, [12](#)
- sySendMulticast
 - Sockets, [23](#)
- sySendSocket
 - Sockets, [20](#)
- sySendSocketAsync
 - Sockets, [20](#)
- sySendToSocket
 - Sockets, [19](#)
- sySetClientSocketOptions
 - Sockets, [19](#)
- sySetDatagramSocketOptions
 - Sockets, [17](#)
- sySetFileInformation
 - Files, [33](#)
- sySetLastNqError
 - Utils, [42](#)
- sySetRand
 - Utils, [41](#)
- sySetSecurityDescriptor
 - Files, [29](#)
- sySetStreamSocketOptions
 - Sockets, [17](#)
- syShutdownSocket
 - Sockets, [17](#)
- sySleep
 - Time, [4](#)
- sySnprintf
 - Input Output, [35](#)
- sySprintf
 - Input Output, [35](#)
- sySscanf
 - Input Output, [35](#)
- sySubscribeToMulticast
 - Sockets, [23](#)
- syThreadDestroy
 - Threads, [8](#)
- syThreadGetCurrent
 - Threads, [8](#)
- syThreadStart

- Threads, [8](#)
- syTruncateFile
 - Files, [31](#)
- syUSleep
 - Time, [4](#)
- syUTF8ToUnicodeN
 - Encoding, [39](#)
- syUnicodeToUTF8N
 - Encoding, [39](#)
- syUnixMode2DosAttr
 - Utils, [41](#)
- syUnlockFile
 - File Locking, [36](#)
- syVsnprintf
 - Input Output, [35](#)
- syWriteFile
 - Files, [31](#)
- syopsyst.h, [43](#)
 - MUTEX_DEBUG, [46](#)
 - SY_OSNAME, [46](#)
 - syAssert, [46](#)
- System Dependent, [3](#)
- Tasks, [24](#)
 - syGetPid, [24](#)
- Threads, [8](#)
 - SYThread, [8](#)
 - syIsValidThread, [8](#)
 - syThreadDestroy, [8](#)
 - syThreadGetCurrent, [8](#)
 - syThreadStart, [8](#)
- Time, [4](#)
 - syComposeTime, [6](#)
 - syConvertTimeInMsecToSec, [6](#)
 - syConvertTimeSpecToTimeInMsec, [4](#)
 - syDecomposeTime, [6](#)
 - syGetTimeInMsec, [4](#)
 - syGetTimeInSec, [4](#)
 - syGetTimeZone, [6](#)
 - syGmtToString, [6](#)
 - sySleep, [4](#)
 - syUSleep, [4](#)
- Utils, [41](#)
 - syGetLastSmbError, [41](#)
 - syRand, [41](#)
 - sySetLastNqError, [42](#)
 - sySetRand, [41](#)
 - syUnixMode2DosAttr, [41](#)